Optimizers Matter in Adversarial Robustness

Hengyue Liang, Buyun Liang, Ying Cui, Tim Mitchell, and Ju Sun

Abstract—Empirical robustness evaluation of neural network models against adversarial perturbations entail solving nontrivial constrained optimization problems. Predominant algorithms rely on numerical methods such as projected gradient and iterative linearization to solve these problems, but require careful hyperparameter tuning to achieve good quality. Moreover, they can only handle ℓ_1, ℓ_2 and ℓ_∞ perturbations due to the use of analytical projectors. In this paper, we introduce a novel algorithmic framework that blends a general-purpose constrained-optimization solver `PyGRANSO`, **W**ith **C**onstraint-**F**olding (PWCF), to add reliability and generality to robustness evaluation. We show that PWCF 1) finds good-quality solutions without the need of delicate hyperparameter tuning, and 2) can handle general perturbation models, e.g., general ℓ_p ($p > 0$) and perceptual attacks, which are inaccessible to existing projector-based algorithms. With the help of this new tool, we further explore the distinct solution patterns found by various combinations of the losses, perturbation models, and optimization algorithm, and discuss the possible implications of these patterns on the robustness research.

Index Terms—deep neural networks, adversarial robustness, adversarial attack, adversarial training, minimal distortion radius, constrained optimization, sparsity, distribution shift

1 Introduction

In visual recognition, deep neural networks (DNNs) are not robust against perturbations that are easily discounted by human perception—either adversarially constructed or naturally occurring [1], [2], [3], [4], [5], [6], [7], [8], [9]. One popular way of finding an adversarial perturbation (a.k.a adversarial attack) is by solving the *adversarial loss* formulation [10], [11]:

$$\max_{\boldsymbol{x}'} \ell(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')) \quad \text{s.t. } \boldsymbol{x}' \in \Delta(\boldsymbol{x}) = \{\boldsymbol{x}' \in [0,1]^n : d(\boldsymbol{x}, \boldsymbol{x}') \le \varepsilon\}, \tag{1}$$

Here, f_θ is the DNN model, and $\Delta(\boldsymbol{x})$ is an allowable perturbation set with radius ε as measured by the metric d. Early works assume $\Delta(\boldsymbol{x})$ is the ℓ_p norm ball intersected with the natural image box, i.e., $\{\boldsymbol{x}' \in [0,1]^n : \|\boldsymbol{x} - \boldsymbol{x}'\|_p \le \varepsilon\}$, where $p = 1, 2, \infty$ are popular choices [2], [11]. To capture visually realistic perturbations, recent works have also modeled nontrivial transformations using non-ℓ_p metrics [3], [4], [5], [6], [7], [8], [9], [12]. Adversarial loss naturally motivates *adversarial training* (AT) as a disciplined framework towards achieving adversarial robustness (AR) [10], [11]:

$$\min_{\boldsymbol{\theta}} \; \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \max_{\boldsymbol{x}' \in \Delta(\boldsymbol{x})} \ell(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')) \tag{2}$$

in contrast to the classical goal of supervised learning:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \ell(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x})) \tag{3}$$

As for empirical robustness evaluation (RE), solutions of Eq. (1) lead to the worst-case perturbations to fool $f_{\boldsymbol{\theta}}$.

- *Hengyue Liang is with the Department of Electrical & Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA. E-mail: liang656@umn.edu.*
- *Buyun Liang and Ju Sun are with the Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, USA. E-mail: {liang664, jusun}@umn.edu.*
- *Ying Cui is with the Department of Industrial & Systems Engineering, University of Minnesota, Minneapolis, MN 55455, USA. E-mail: yingcui@umn.edu.*
- *Tim Mitchell is with the Department of Computer Science,Queens College, City University of New York, Flushing, NY 11367, USA. E-mail: tmitchell@qc.cuny.edu.*

But solving Eq. (1) is not easy: the objective is non-concave for typical choices of loss ℓ and model $f_{\boldsymbol{\theta}}$; for non-ℓ_p metrics d, $\Delta(\boldsymbol{x})$ is often a complicated nonconvex set. In practice, there are two major lines of algorithms: **(a) direct numerical maximization** that takes differentiable ℓ and $f_{\boldsymbol{\theta}}$, and tries direct maximization, e.g., using gradient-based methods [11], [13]. This often only produces a suboptimal solution and can lead to overoptimistic RE; **(b) upper-bound maximization** that constructs tractable upper bounds for the margin loss $\ell_{\text{ML}} = \max_{i \ne y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') - f^y_{\boldsymbol{\theta}}(\boldsymbol{x}')$ (where y is the true class of $\boldsymbol{x}$ and the boldface $\boldsymbol{y}$ is the one-hot encoding of y) and then optimizes against the upper bounds [14]. Improving the tightness of upper bounding while maintaining tractability is still an active area of research. It is worth mentioning that since $\ell_{\text{ML}} \ge 0$ implies an attack success, family **(b)** is also often used to verify the robustness of $f_{\boldsymbol{\theta}}$ over $\Delta(\boldsymbol{x})$ by providing an underestimate of the robust accuracy [14], [15], [16], [17], [18], [19]; through careful construction they can also be used for AT [20], [21], [22], [23], [24].

Another formalism of robustness is the *robustness radius* (or minimum distortion radius), defined as the minimal level of perturbation that causes $f_{\boldsymbol{\theta}}$ to change its predicted class:

$$\min_{\boldsymbol{x}' \in [0,1]^n} d(\boldsymbol{x}, \boldsymbol{x}') \quad \text{s.t. } \max_{i \ne y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') \ge f^y_{\boldsymbol{\theta}}(\boldsymbol{x}'). \tag{4}$$

Solving Eq. (4) produces not only a minimal perturbation $\boldsymbol{x}'$, but also a robustness radius, making Eq. (4) another popular choice for RE[1] [13], [26], [27]. For small and restricted $f_{\boldsymbol{\theta}}$ and selected d, Eq. (4) can be solved exactly by mixed integer programming [28], [29], [30]. For general $f_{\boldsymbol{\theta}}$ and selected d, lower bounds of the robust radius can be computed [31], [32], [33], [34]. But in general, Eq. (4) is heuristically solved via gradient-based methods or iterative linearization [1], [26], [27], [35], [36], [37], [38] for upper bounds, see Section 2.

Our contributions In this paper, we first focus on numerical optimization of Eq. (1) and Eq. (4), and then study the

1. One can also perform AT using Eq. (4) via bi-level optimization; see, e.g., [25].

implications on RE. As for solving Eq. (1) and Eq. (4), we **(I)** adapt the constrained-optimization solver `PyGRANSO` [39], [40] **W**ith **C**onstraint-**F**olding (PWCF) technique—crucial for making `PyGRANSO` solve Eq. (1) and Eq. (4) with reasonable speed and quality, and **(II)** show that PWCF can handle distance metric d *other than the popular but unrealistic* ℓ_1, ℓ_2, *and* ℓ_∞ *ones*—beyond the reach of PGD-based methods. This can lead to considerably improved RE as PWCF **(I)** can serve as a *reliable supplement* to the state-of-the-art (SOTA) RE packages on ℓ_1, ℓ_2, and ℓ_∞ attacks, e.g. `AutoAttack` [13], and **(II)** opens up the possibility of RE over a much wider range of adversarial perturbation models, e.g., general ℓ_p attacks with any $p > 0$ and more complicated ones such as perceptual attacks [12]. Then we highlight the complex interplay between loss ℓ, perturbation set $\Delta(\boldsymbol{x})$, and numerical methods for solving Eq. (1) and Eq. (4), and we observe that these factors cause distinct patterns in the solutions. We identify these patterns as a major challenge facing RE.

2 Technical background

Numerical maximization of Eq. (1) Eq. (1) is often solved by the projected gradient descent (PGD)[2] method. The basic update reads $\boldsymbol{x}'_{new} = \mathcal{P}_{\Delta(\boldsymbol{x})}(\boldsymbol{x}'_{old} + t\nabla\ell(\boldsymbol{x}'_{old}))$, where $\mathcal{P}_{\Delta(\boldsymbol{x})}$ is the projection operator onto $\Delta(\boldsymbol{x})$. When $\Delta(\boldsymbol{x}) = \{\boldsymbol{x}' \in [0,1]^n : \|\boldsymbol{x}' - \boldsymbol{x}\|_p \le \varepsilon\}$ with $p = 1, \infty$, $\mathcal{P}_{\Delta(\boldsymbol{x})}$ takes simple forms. For $p = 2$, sequential projection onto the box and then the norm ball at least finds a feasible solution (see our clarification of these projections in Section A). Hence PGD is feasible for these cases. For *other choices of* p *and general non-*ℓ_p *metrics* d where analytical projection is not so intuitive to derive, existing PGD based algorithms does not apply. For practical PGD methods, previous works have shown that the solution quality is sensitive to the tuning of multiple hyperparameters, e.g., step-size schedule and iteration budget [13], [41], [42]. The SOTA PGD variants, APGD-CE and APGD-DLR, try to make the tuning automatic by combining a heuristic adaptive step-size schedule and momentum acceleration under fixed iteration budget [13]—both are built into the popular `AutoAttack` package[3]. Another non-PGD method for Eq. (1), Square Attack [43], which is based on gradient-free random search is also included in `AutoAttack`, but its performance is not as effective as PGD based ones.

Numerical minimization of Eq. (4) The essential difficulty in optimizing Eq. (4) lies at dealing with the highly nonlinear constraint $\max_{i \neq y} f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') \ge f^y_{\boldsymbol{\theta}}(\boldsymbol{x}')$. There are two lines of ideas to circumvent it: **(1) penalty methods** turn the constraint into a penalty term added to the objective [1], [37]. The resulting box-constrained problems can then be handled by classical optimization methods, such as L-BFGS [44] or PGD. But penalty methods are not guaranteed to return feasible solutions for the original problem; **(2) iterative linearization** linearizes the nonlinear constraint in each step, leading to simple solutions to the projection (onto the intersection of the linearized hyperplane and the $[0,1]^n$ box) for easy choices of d (e.g., ℓ_p distances); see, e.g., [26], [35]. But again, for general d, the projection is unlikely to be derived easily. In [27], [38], the problem $\min_{\boldsymbol{x}'} d(\boldsymbol{x}, \boldsymbol{x}')$ is reformulated into $\min_{\boldsymbol{x}', t}\ t,\ \text{s.t.}\ d(\boldsymbol{x}, \boldsymbol{x}') \le t$ so that the perturbation direction (determined by $\boldsymbol{x}'$) and the norm (determined by t) are decoupled, and then penalty methods and iterative linearization are combined. In the SOTA RE package `AutoAttack`, the included fast adaptive boundary (FAB) attack [26] belongs to family **(2)**.

`PyGRANSO` for constrained optimization As stated above, most of the above methods depend on simple projectors and do not work when d is sophisticated for the constrained optimization problem Eq. (1) and Eq. (4); previous works [13], [41], [42] also have shown that the solution quality using these handcrafted methods is sensitive to hyperparameters tuning: e.g., step-size schedule and iteration budget. Furthermore, the diversity of solutions for Eq. (1) and Eq. (4) plays a critical role for RE (which is highlighted in [13], [42], and we will also share our thought in Section 5). All of these issues underscore the need for alternative solvers.

In principle, as an instance of general nonlinear optimization (NO) problems [44], [45]

$$\min_{\boldsymbol{x}}\ g(\boldsymbol{x}) \\ \text{s.t. } c_i(\boldsymbol{x}) \le 0\ \forall\, i \in \mathcal{I};\ h_j(\boldsymbol{x}) = 0\ \forall\, j \in \mathcal{E}, \tag{5}$$

Eq. (1) and Eq. (4) can be solved by general-purpose NO solvers such as `Knitro` [46], `Ipopt` [47], and `GENO` [48], [49]. However, there are two caveats: (1) these solvers only handle continuously differentiable objective and constraint functions, i.e., g and all c_i's and h_j's. But non-differentiable g and c_i's and h_j's are common in Eq. (1) and Eq. (4), e.g., when d is the ℓ_1 and ℓ_∞ distance, or $f_{\boldsymbol{\theta}}$ uses non-differentiable activations; (2) they require analytical gradients of g, c_i's, and h_j's, which are impractical to derive when DNN models $f_{\boldsymbol{\theta}}$ are involved.

`PyGRANSO`[4] [39], [40] is a recent `PyTorch`-port of the powerful MATLAB package `GRANSO` [39] which can handle general NO problems of form Eq. (5) with non-differentiable g, c_i's, and h_j's. It only requires these functions to be *almost everywhere differentiable* [50], [51], [52] which are satisfied by almost all forms of Eq. (1) proposed so far in the literature. `GRANSO` employs a quasi-Newton sequential quadratic programming (BFGS-SQP) to solve Eq. (5), and features a rigorous adaptive step-size rule via line search and a principled stopping criterion inspired by gradient sampling [53]. See a sketch of the algorithm in Section B and [39] for details. `PyGRANSO` equips `GRANSO` with auto-differentiation and GPU computing powered by `PyTorch`—crucial for deep learning problems. The stopping criterion is controlled by stationarity, total constraint violation, and optimization tolerance—all can be transparently controlled, but is typically unnecessary to tune. For the details of `PyGRANSO` package, please check: https://arxiv.org/abs/2210.00973.

Min-max optimization of Eq. (2) For a finite training set, Eq. (2) becomes

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \max_{\boldsymbol{x}'_i \in \Delta(\boldsymbol{x}_i)} \ell(\boldsymbol{y}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}'_i)) \tag{6}$$

2. It should be "ascent" instead of "descent" due to the maximization, but we follow the `AutoAttack` package.

3. https://github.com/fra31/auto-attack

4. https://ncvx.org

$$\iff \min_{\boldsymbol{\theta}} \max_{\boldsymbol{x}_i' \in \Delta(\boldsymbol{x}_i)\ \forall i} \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{y}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i')) \tag{7}$$

i.e., in the form of $\min_{\boldsymbol{\theta}} \max_{\boldsymbol{x}_i' \in \Delta(\boldsymbol{x}_i)\ \forall i} \phi(\boldsymbol{\theta}, \{\boldsymbol{x}_i\})$. So Eq. (6) is often solved by iterating between the (separable) inner maximization and a subgradient update for the outer minimization. The latter takes a subgradient of $h(\boldsymbol{\theta}) \doteq \max_{\boldsymbol{x}_i' \in \Delta(\boldsymbol{x}_i)\ \forall i} \phi(\boldsymbol{\theta}, \{\boldsymbol{x}_i\})$ from the subdifferential $\partial_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}, \{\boldsymbol{x}_i^*\})$ ($\boldsymbol{x}_i^*$ are maximizers for the inner maximization), justified by the celebrated Danskin's theorem [54], [55], [56]. However, if the numerical solutions to the inner maximization are substantially suboptimal[5], the subgradient we take can be misleading — see our discussion in Section C. Thus, solving Eq. (7) requires high-quality solutions for Eq. (1).

3 PyGRANSO with Constraint Folding as a Generic Solver for Eq. (1) and Eq. (4)

Though `PyGRANSO` can serve as a promising solver for Eq. (1) and Eq. (4) with general metric d, we find in practice that naive deployment can suffer from slow convergence, or low quality solutions due to numerical issues. Below, we introduce `PyGRANSO` **W**ith **C**onstraint-**F**olding (PWCF) and other techniques, which can substantially speed up the optimization process and improve the solution quality.

3.1 Reformulating ℓ_∞ constraint to avoid sparse subgradients

The BFGS-SQP algorithm inside `PyGRANSO` relies on the subgradients of the objective and the constraint functions to approximate the (inverse) Hessian and to compute the search direction. Hence, when the subgradients are sparse, updating all optimization variables may take many iterations, leading to slow convergence. For the ℓ_∞ metric,

$$\partial_{\boldsymbol{z}} \|\boldsymbol{z}\|_\infty = \operatorname{conv}\{\boldsymbol{e}_k \operatorname{sign}(z_k) : z_k = \|\boldsymbol{z}\|_\infty\ \forall k\}, \tag{8}$$

where $\boldsymbol{e}_k$'s are the standard basis vectors, conv denotes convex hull, and $\operatorname{sign}(z_k) = z_k/|z_k|$ if $z_k \neq 0$, else $[-1, 1]$. The subgradient in Eq. (8) contains no more than $n_k = |\{k : z_k = \|\boldsymbol{z}\|_\infty\}|$ nonzeros, and hence is sparse when n_k is small. To avoid this issue, we propose a reformulation

$$\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \le \varepsilon \iff -\varepsilon \mathbf{1} \le \boldsymbol{x} - \boldsymbol{x}' \le \varepsilon \mathbf{1}, \tag{9}$$

where $\mathbf{1} \in \mathbb{R}^n$ is the all-one vector.

3.1.1 Constraint-folding to reduce the number of constraints

The natural image constraint $\boldsymbol{x}' \in [0,1]^n$ is a set of n box constraints. The reformulation described in Section 3.1 introduces another $\Theta(n)$ box constraints. Although all these are just simple linear constraints, the growth of $\Theta(n)$ is daunting: for natural images, n is the number of pixels that can easily get into hundreds of thousands. Typical NO problems become more difficult when the number of constraints grows, e.g., leading to slow convergence for numerical algorithms.

5. [11] argues using numerical evidence that the maximization landscapes are typically benign in practice and hence gradient-based methods often find solutions with function values close to the global optimal.

To combat this, we introduce a folding technique that can reduce the number of constraints into a small constant. To see how this is possible, first note that any equality constraint $h(\boldsymbol{x}) = 0$ or inequality constraint $c(\boldsymbol{x}) \le 0$ can be reformulated as

$$\begin{aligned} h(\boldsymbol{x}) = 0 &\iff |h(\boldsymbol{x})| \le 0\,, \\ c(\boldsymbol{x}) \le 0 &\iff \max\{c(\boldsymbol{x}), 0\} \le 0. \end{aligned} \tag{10}$$

We can fold them together as

$$\mathcal{F}(|h(\boldsymbol{x})|, \max\{c(\boldsymbol{x}), 0\}) \le 0 \tag{11}$$

where $\mathcal{F} : \mathbb{R}_+^2 \mapsto \mathbb{R}_+$ ($\mathbb{R}_+ \doteq \{t : t \ge 0\}$) can be any function satisfying $\mathcal{F}(\boldsymbol{z}) = 0 \implies \boldsymbol{z} = \mathbf{0}$, e.g., any ℓ_p ($p \ge 1$) norm.

It is easy to verify the equivalence of Eq. (11) and the original constraints in Eq. (10). The folding technique can be used to a subset or all of the constraints; one can group and then fold constraints according to their physical meanings. We note that folding or aggregating constraints is not a new idea and has been popular in engineering design. E.g., [59] uses ℓ_∞ folding and its log-sum-exponential approximation to deal with numerous design constraints; also see [60], [61], [62], [63]. However, applying folding into NO problems in machine learning seems rare, potentially because producing non-differentiable constraint(s) due to the folding seems counterproductive.

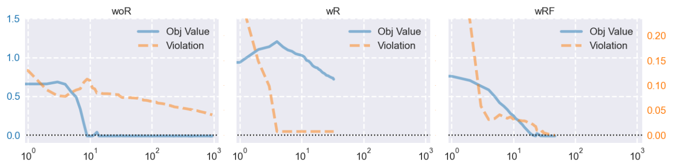


Fig. 1. Optimization trajectory of the **objective value** and **constraint violation** w.r.t iterations for an ℓ_∞ case on CIFAR-10 dataset. **woR**: using ℓ_∞ original form; **wR**: with reformulation but no folding; **wRF**: with reformulation and folding. Maximum time budget per curve: $600s$ (only wRF terminates before reaching this budget). Both **objective** and **violation** reaching 0 indicates successful attack.

In our experiments, we use $\mathcal{F} = \|\cdot\|_2$ to fold the $\Theta(n)$ box constraints from ℓ_∞ reformulation into a single constraint, enforce the $\boldsymbol{x}' \in [0,1]^n$ constraints in $f_{\boldsymbol{\theta}}$ by direct clipping. Fig. 1 shows clearly that combining folding and reformulation can substantially speed up convergence and boost the solution quality for our algorithm.

3.1.2 Reformulation of Eq. (4) for ℓ_∞ norm

When solving Eq. (4) by `PyGRANSO` with ℓ_∞ norm as the objective function, we reformulate the problem as:

$$\begin{aligned} \min_{t, \boldsymbol{x}' \in [0,1]^d} \quad & t \\ \text{s. t.} \quad & \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \le t \\ & \max_{i \neq y} f_{\boldsymbol{\theta}}^i(\boldsymbol{x}') \ge f_{\boldsymbol{\theta}}^y(\boldsymbol{x}') \end{aligned} \tag{12}$$

By introducing the dummy variable t, the ℓ_∞ objective function is moved into the constraints, and the folding technique Eq. (9) and Eq. (11) can thus be applied naturally to avoid the sparse subgradients and speed up the optimization process.

TABLE 1
Comparison of our PWCF with SOTA attack methods on ℓ_1-, ℓ_2- and ℓ_∞- attacks. For given pretrained models, we report the models' **clean** and **robust** accuracy—lower **robust** accuracy means more effective attacks. We test on both CE and margin loss for APGD and PWCF. Numbers are in $(\%)$. **Model - Attack** denotes the selection of the models and the type of the performed adversarial attacks and its ε.

			APGD		PWCF (ours)		Square	APGD+
Dataset	**Model - Attack**	**Clean**	**CE**	**M**	**CE**	**M**	**M**	**PWCF**
CIFAR10	P$_1$ [57] - $\ell_1(12)$	73.3	0.96	0.00	28.6	0.00	2.28	**0.00**
	WRN$_{\text{-70-16}}$ [58] - $\ell_2(0.5)$	94.7	81.8	81.1	81.8	81.0	87.9	**80.8**
	WRN$_{\text{-70-16}}$ [58] - $\ell_\infty(0.03)$	90.8	69.4	68.0	73.6	72.8	71.6	**67.1**
ImageNet100	PAT$_{\text{-Alex}}$ [12] - $\ell_2(4.7)$	75.0	42.7	44.0	42.8	44.5	63.1	**40.9**
	PAT$_{\text{-Alex}}$ [12] - $\ell_\infty(0.016)$	75.0	48.0	48.2	56.6	48.8	59.9	**45.2**

TABLE 2
Comparison of our PWCF with SOTA minimal radius method (FAB) on ℓ_1-, ℓ_2- and ℓ_∞- attacks. For given pretrained models, we report the models' minimal perturbation radius—lower radius means more effective minimization. We test on both CE and margin loss for APGD and PWCF. Numbers are in $(\%)$. **Model - Distance** denotes the selection of the models and the distance type of the performed minimal perturbation.

		FAB			PWCF (ours)			PWCF - FAB		
Dataset	**Model - Distance**	**Mean**	**Median**	**STD**	**Mean**	**Median**	**STD**	**Mean**	**Median**	**STD**
CIFAR10	PAT$_{\text{-Self}}$ [12] - ℓ_1	13.92	10.50	12.63	12.02	7.290	11.46	**-1.889**	**-0.809**	5.238
	PAT$_{\text{-Self}}$ [12] - ℓ_2	1.016	0.8950	0.7095	0.9973	0.8817	0.6907	**-0.01890**	**-0.01508**	0.2504
	PAT$_{\text{-Self}}$ [12] - ℓ_∞	0.02978	0.02448	0.02199	0.02978	0.02521	0.02235	**0.0007800**	**-0.00007296**	0.007053
ImageNet100	PAT$_{\text{-Alex}}$ [12] - ℓ_1	435.4	400.6	303.9	408.1	390.6	284.7	**-27.31**	**-13.46**	70.55
	PAT$_{\text{-Alex}}$ [12] - ℓ_2	6.747	6.812	3.813	6.705	6.876	3.758	**-0.04193**	**-0.03546**	0.7575
	PAT$_{\text{-Alex}}$ [12] - ℓ_∞	0.02801	0.02826	0.01562	0.02891	0.02903	0.01627	**0.0008936**	**0.00001500**	0.002414

3.2 Loss clipping when solving Eq. (1) with PWCF

For Eq. (1) with the popular cross-entropy (CE) and margin losses, the objective value can easily dominate constraint violation during the maximization process. Since `PyGRANSO` tries to balance the objective value and constraint violation when making progress, it can persistently prioritize optimizing the objective over constraint satisfaction, resulting in very slow progress in finding a feasible solution. To resolve this numerical difficulty, we propose using clipped margin loss ℓ_{ML} with maximal value 0.01, as any $\ell_{ML} \geq 0$ indicates a successful attack. For the same reason, we use clipped CE loss with maximal value at 10 in PWCF[6].

4 Experiments and Results: Solving Eq. (1) and Eq. (4) with PWCF.

In this section, we demonstrate the solution quality of using PWCF to solve the RE problem Eq. (1) and Eq. (4). In section 4.1 and 4.2, we compare PWCF with the SOTA methods w.r.t ℓ_1, ℓ_2 and ℓ_∞ distance respectively. In section 4.3, we demonstrate the solution quality of PWCF w.r.t other general distances.

6. Attack success happens when the true logit output less than $1/K$ (assuming softmax normalization is applied), where K is the number of classes. So the critical value is $-\log 1/K$, which is < 10 for $K \leq e^{10}$, sufficient for typical RE datasets.

4.1 PWCF offers competitive and complementary attack performance to Eq. (1)

We take SOTA ℓ_1-, ℓ_2-, and ℓ_∞-adversarially trained models on CIFAR10[7][8], and an adversarially-trained model with respect to the LPIPS distance[9] on ImageNet [12][10], to compare the attack performance by solving Eq. (1) between PWCF and the APGD[11] [64] method from `AutoAttack` package. The attack bound ε are set following the common practice of adversarial RE[12].

From Table 1, we can conclude that: (1) PWCF performs strongly and comparably to APGD on ℓ_1, ℓ_2 and ℓ_∞ attacks, especially using *margin loss* as the objective; (2) PWCF is weak on ℓ_1 and ℓ_∞ attacks using CE loss, likely due to the bad numerical scaling of the CE loss; (3) Combining all successful attack samples found by APGD and PWCF (APGD+PWCF) can further reduce the robust accuracy compared to any single APGD or PWCF attack—PWCF and APGD are complementary. Note that [42] also remarks that the diversity of solutions matters much more than the superiority of individual solvers, which is the reason why `AutoAttack` includes Square Attack–a zero-th order black-box attack method that does not perform strongly itself as

7. https://github.com/locuslab/robust_union/tree/master/CIFAR10
8. https://github.com/deepmind/deepmind-research/tree/master/adversarial_robustness
9. See Section 4.3 for details.
10. https://github.com/cassidylaidlaw/perceptual-advex
11. We implement the margin loss on top of `AutoAttack`.
12. E.g., https://robustbench.github.io/ for Cifar10 ℓ_2 and ℓ_∞; https://github.com/locuslab/robust_union for Cifar10 ℓ_1; [12] for ImageNet ℓ_2 and ℓ_∞.

shown in Table 1. Our Section 5 below contains expansive discussion on the patterns of solutions found by different methods and the practical implications.

4.2 PWCF provides competitive minimal radius solutions to Eq. (4)

4.2.1 Benefits of Eq. (4) *over Eq.* (1)

Currently, reasons to select the attack budget ε in RE, as far as we know, is purely empirical, and thus debatable. E.g., **Reason 1**: some argues that setting $\varepsilon = 0.5$ for Cifar10 dataset in Eq. (1) with ℓ_2 distance, is that this budget is sufficient to fool a model without adversarial training on all images, while adversarially trained models cannot achieve 100% accuracy — can be used to distinguish robustness performance in RE. However, when comparing model robustness, better performance on a fix ε cannot guarantee a better one when ε varies — conclusion drawn on a single ε can be misleading. **Reason 2:** others argue that such ε choice is to keep the perturbation "imperceptible". This argument also cannot justify 1) why slightly larger values is not imperceptible as it is a subjective measure by human and 2) real-life perturbations can go way beyond imperceptible level and is natural.

On the contrary, solving Eq. (4) will not have the above problems: the solutions are sample-specific and the distance $d(\boldsymbol{x}, \boldsymbol{x}')$ found is different (and minimal) for every sample $\boldsymbol{x}$ — this can provide much more statistical information for RE, especially useful in comparing model robustness. From Table 2 and Fig. 4, we can conclude that: (1) PWCF performs strongly and comparably to FAB on the robust radius solutions of ℓ_1, ℓ_2 and ℓ_∞ norms, especially using ℓ_2 and ℓ_∞ norms as the distance.

4.3 PWCF works for general (almost everywhere) differentiable ℓ_p and non-ℓ_p distances to Eq. (1) and Eq. (4)

As highlighted in Section 2, a major limitation of the PGD based solvers is that they cannot handle distances other than ℓ_1, ℓ_2, and ℓ_∞[13]. By contrast, PWCF stands out as a convenient choice for general distances. To show this, we apply PWCF to solve both Eq. (1) and Eq. (4) with $\ell_{1.5}$ and ℓ_8 distances. As far as we know, there are no prior works dealing with general ℓ_p distances, and hence no reference in choosing ε for Eq. (1). Therefore, we carry out the following experiment: 1) First, we solve Eq. (4) using PWCF and record the minimal radius $d_i(\cdot)$ of each image I_i under $\ell_{1.5}$ and ℓ_8 distances; see Fig. 2 and Fig. 3. 2) We solve Eq. (1) using $\varepsilon_i = d_i(\cdot)$ found in 1) for every image I_i, and report the attack success rate of the dataset with the per-image bound ε_i. We take the `clean` and `pat_alexnet_0.5` models from [12] respectively as examples, which are both trained on ImageNet100 dataset. In addition, we also consider solving Eq. (1) with the perceptual adversarial threat (PAT) model [12], where the distance is defined using the popular perceptual metric LPIPS [66]:

$$
\begin{aligned}
& d(\boldsymbol{x}, \boldsymbol{x}') \doteq ||\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')||_2 \\
\text{where} \quad & \phi(\boldsymbol{x}) \doteq [\, \widehat{g}_1(\boldsymbol{x}), \dots, \widehat{g}_L(\boldsymbol{x}) \,]
\end{aligned} \tag{13}
$$

13. We do not consider ℓ_0 in this paper as it is not a norm, but we acknowledge that [65] targets at generating ℓ_0 attacks using PGD-based method.

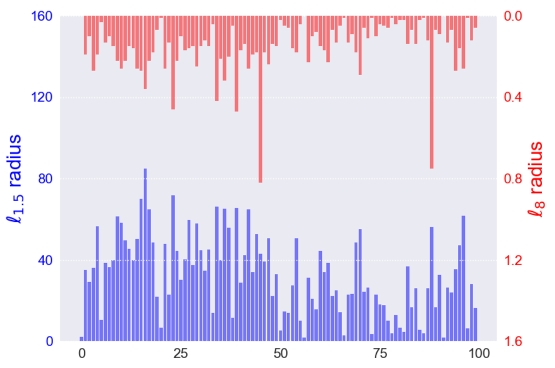


Fig. 2. Minimal robust radius found by solving Eq. (12) using PWCF with $\ell_{1.5}$ and ℓ_8.

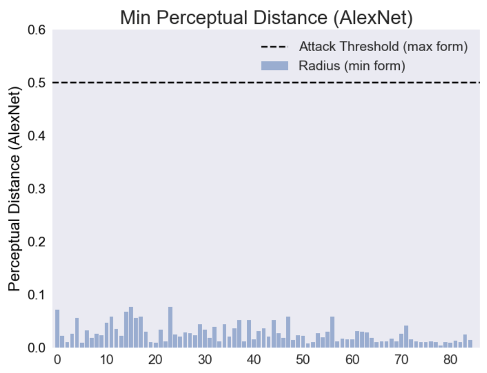


Fig. 3. Minimal robust radius found by solving Eq. (12) using PWCF with perceptual distance.

where $\widehat{g}_1(\boldsymbol{x}), \dots, \widehat{g}_L(\boldsymbol{x})$ are the vectorized intermediate feature maps from pretrained DNNs.

TABLE 3
Attack performance of PWCF with margin loss on general ℓ_p and non-ℓ_p metrics. We report attack success rates (numbers are in %). We test on $\ell_{1.5}$, ℓ_8, and PAT; numbers on ℓ_1, ℓ_2, and ℓ_∞ are included for reference. Numbers below each rate in parenthesis are the perturbation radii.

	Special ℓ_p			General ℓ_p		
Model	ℓ_1	ℓ_2	ℓ_∞	$\ell_{1.5}$	ℓ_8	PAT
Clean	100	100	100	100	100	100
	(2361)	(6.09)	(0.01569)	(44.40)	(0.07)	(0.5)
PAT	49.7	40.7	35.2	100	100	100
	(2400)	(4.7)	(0.017)	(443.98)	(0.70)	(0.5)

PWCF handles them seamlessly, as shown in Table 3. Here we do not strive to set the most reasonable perturbation radii, especially for $\ell_{1.5}$ and ℓ_8 that have not been tested before, and hence we also do not stress the attack rates. Our point is that *PWCF is able to handle these general ℓ_p distances*. Table 4 further summarizes the details of performing the perceptual attack with $\varepsilon = 0.5$. Existing methods to compare are Perceptual Projected Gradient Descent (PPGD), Lagrangian perceptual attack (LPA) and its variant fast Lagrangian perceptual attack (Fast-LPA) methods, all developed in [12], based on iterative linearization and projection (PPGD), or penalty method (LPA, Fast-LPA) respectively. In addition to the objective values and attack success rates, we also report their chances of finding

TABLE 4
Performance comparison of different methods solving PAT with the clipped CE and margin (M) loss. **Viol.** reports the ratio of final solutions that violate constraints. **Succ.** is the ratio of all *feasible successful attacks* divided by *total number of samples*. The model we test is `pat_alexnet_0.5` [12]. Evaluation is performed on ImageNet-100 dataset.

	CE Objective		Margin Objective	
Method	**Viol. (%)** $\downarrow$	**Att. Succ. (%)** $\uparrow$	**Viol. (%)** $\downarrow$	**Att. Succ. (%)** $\uparrow$
Fast-LPA	73.8	3.54	41.6	56.8
LPA	**0.00**	80.5	**0.00**	97.0
PPGD	5.44	25.5	**0.00**	38.5
PWCF	0.62	93.6	**0.00**	100

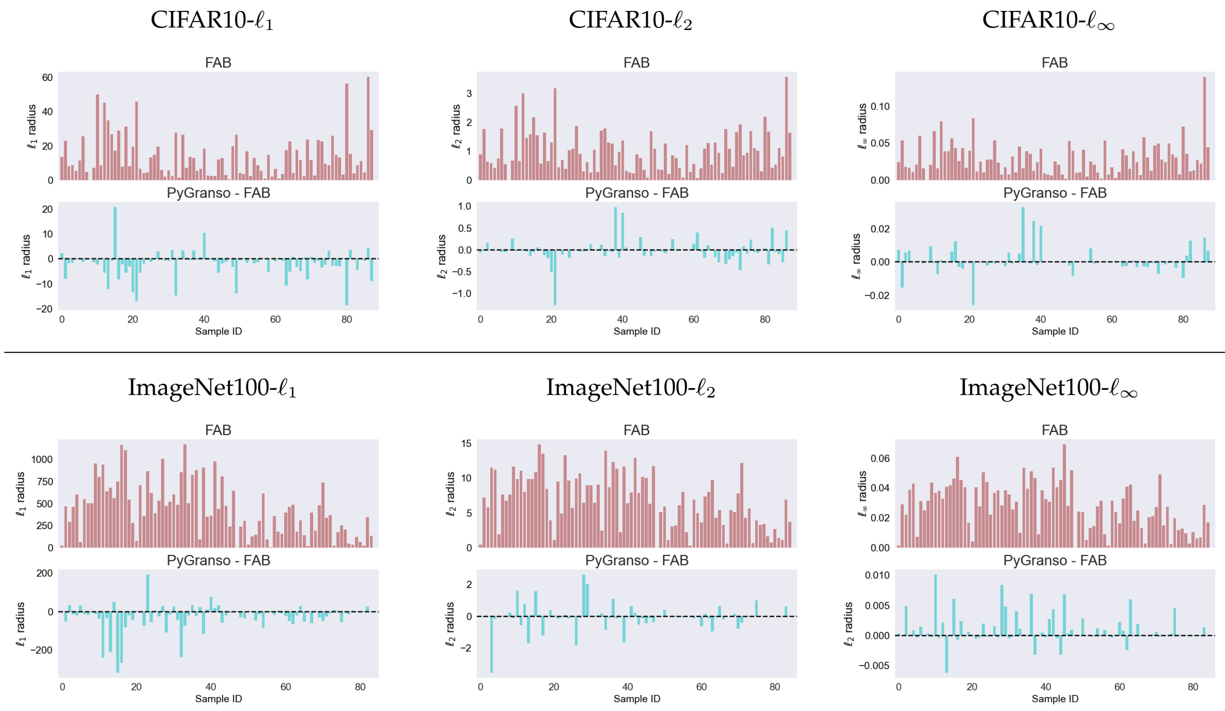


Fig. 4. Minimal robust radius found by solving Eq. (12) using FAB and PWCF with ℓ_1, ℓ_2 and ℓ_∞. **1st row**: minimal robust radius of each CIFAR-10 attack sample found by FAB; **2nd row**: difference of minimal robust radius of each CIFAR-10 attack sample found by PWCF and FAB (negative value means PWCF performs better); **3rd row**: minimal robust radius of each ImageNet100 attack sample found by FAB; **4th row**: difference of minimal robust radius of each CIFAR-10 attack sample found by PWCF and FAB (negative value means PWCF performs better).

infeasible solutions. As observed in Table 4, our PWCF is the clear winner.

5 Difficulty of Achieving AR via AT in Eq. (2)

5.1 Approximate global maximizers of Eq. (1) have different patterns

[11] attributes the empirical success of AT for AR to the surprisingly effective global maximization of Eq. (1) using gradient-based methods in practice—thanks to benign maximization landscapes. Moreover, it observes that the (approximate) global maximizers are distinct and spatially scattered. Here, we go one step further and show that these global maximizers have different patterns.

To show why this may happen, we take the ℓ_1 distance, i.e., $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 \leq \varepsilon$ and ignore the box constraint. The ℓ_1 norm is a famous sparsity-promoter in statistics and machine learning [67], [68], and hence we expect the estimated perturbation $\boldsymbol{\delta} = \boldsymbol{x}' - \boldsymbol{x}$ to be sparse. For simplicity, we take the loss as $0/1$ classification error $\ell(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')) =$

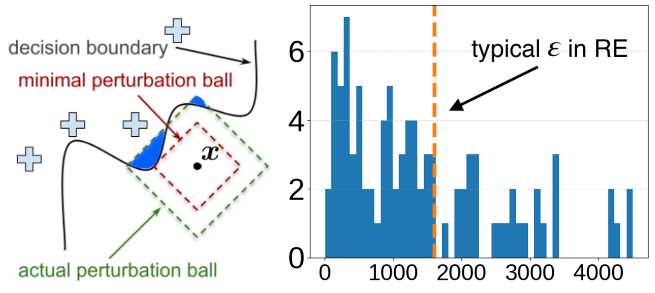


Fig. 5. **(left)** Geometry of Eq. (1) with multiple global maximizers. Here we consider the ℓ_1-norm ball around $\boldsymbol{x}$, and ignore the box constraint $\boldsymbol{x}' \in [0,1]^n$ that typically does not substantially change the geometry. Depending on the loss ℓ used, part or the whole of the blue regions becomes global or near-global maximizers; **(right)** Histogram of the ℓ_1 robustness radii estimated by solving Eq. (4) for 100 randomly selected images from ImageNet100. The typical ℓ_1 perturbation radius that people use in RE is close to 1600.

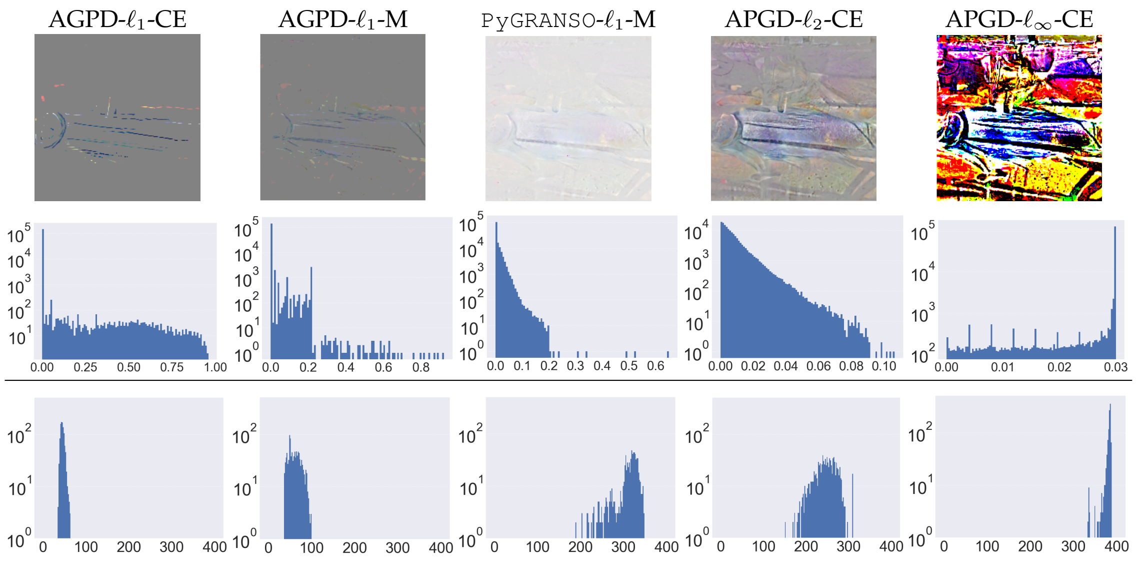


Fig. 6. Sparsity patterns of the perturbations found by different algorithm-distance-loss combinations when attacking the `pat_alexnet_0.5` model [12] using Eq. (1). **1st row**: different perturbations $\boldsymbol{\delta}$'s on a fish image; **2nd row**: histograms of pixels inside each perturbation $\boldsymbol{\delta}$; **3rd row**: histograms of the sparsity measure $\|\boldsymbol{\delta}\|_1/\|\boldsymbol{\delta}\|_2$ over 1024 images randomly selected from ImageNet100. Here, CE: cross-entropy loss; M: margin loss, and we set $\varepsilon = 1000$ for all ℓ_1 attacks, $\varepsilon = 4.6$ for all ℓ_2, and $\varepsilon = 0.03$ for all ℓ_∞, respectively.

$\mathbb{1}\left\{\max_i f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') \neq y\right\}$. Note that ℓ is maximized whenever $f^i_{\boldsymbol{\theta}}(\boldsymbol{x}') > f^y_{\boldsymbol{\theta}}(\boldsymbol{x}')$ for a certain $i \neq y$, so that $\boldsymbol{x}'$ crosses the local decision boundary between the i-th and y-th classes; see Fig. 5(left). In practice, people set a substantially larger perturbation radius than the robustness radius—which can be estimated by solving Eq. (4); see Fig. 5(right). Thus, there could be infinitely many global maximizers (the shaded blue regions in Fig. 5(left)), depending on the shape of the decision boundary. More interestingly, the optimal perturbations can have different sparsity patterns, leading to different patterns in these global maximizers. Now for general losses, such as the cross-entropy and the margin loss, the set of global maximizers might be smaller, but the patterns in them also tend to be much more complicated due to the typical complicated nonlinear decision boundaries associated with DNN models.

5.2 Different combinations ℓ, d, and solvers prefer different patterns

Contrary to the ℓ_1 norm that induces sparsity, the ℓ_∞ norm promotes dense perturbations with comparable entry-wise magnitudes [69] and the ℓ_2 norm promotes dense perturbations whose entries follow power-law distributions. These varying sparsity patterns are evident from Fig. 6, when we compare the first three columns that work with the ℓ_1 norm with the last two columns that work with ℓ_2 and ℓ_∞ norms, respectively. But, we highlight the fine-grained patterns induced by the loss ℓ and numerical solvers on top of these:

- **Margin loss induces denser patterns than cross-entropy** Columns 1 and 2 of Fig. 6 show this for APGD with the ℓ_1 distance on CE and M losses, respectively. For the single perturbation found for the "fish" image, APGD-ℓ_1-M contains numerous nonzeros concentrated in $(0, 0.2]$ compared to the relatively few nonzeros produced by APGD-ℓ_1-CE spread over $(0, 1]$. Moreover, the sparsity (measured by the scale-invariant metric ℓ_1/ℓ_2—larger the denser) statistics of perturbations over randomly selected 1024 images from ImageNet100 (3-rd row of Fig. 6) confirms the trend.
- **`PyGRANSO` induces denser patterns than APGD** For ℓ_1 distance with M, when we switch the solver from APGD to `PyGRANSO` we also observe substantially denser perturbation patterns; see Columns 2 and 3 of Fig. 6. In fact, the histograms on the 3-rd row show that the quantitative sparsity levels are clearly separated.

These trends are also observed on other norms. While we cannot coherently explain how the fine-grained patterns are induced by the complex interplay of ℓ, d, and the numerical solver, their very presence seems prevalent.

6 Conclusion

In this paper, we propose PWCF to solve the maximization problem Eq. (1) and minimization problem Eq. (4) in robustness evaluations, blending the SOTA constrained optimization solver `PyGRANSO` with constraint folding and other tweaks. Our experimental results show that 1) PWCF can provide competitive and complementary performance compared with the SOTA methods on ℓ_1, ℓ_2, and ℓ_∞ attacks; 2). PWCF can provide competitive performance compared with the SOTA methods on minimization of ℓ_1, ℓ_2, and ℓ_∞ distances. 3) PWCF can deal with general attack models such as ℓ_p with $p \geq 1$ and perceptual attacks and their corresponding minimal robust radius problem, which are beyond the reach of existing PGD-based methods; 4) PWCF involves little to zero parameter-tuning and obtains reliable solutions based on a principled stopping criterion.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[3] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations*, 2018.

[4] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1802–1811. [Online]. Available: https://proceedings.mlr.press/v97/engstrom19a.html

[5] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *International Conference on Learning Representations*, 2018.

[6] E. Wong, F. R. Schmidt, and J. Z. Kolter, "Wasserstein adversarial examples via projected sinkhorn iterations," *arXiv:1902.07906*, Feb. 2019.

[7] C. Laidlaw and S. Feizi, "Functional adversarial attacks," *Advances in neural information processing systems*, vol. 32, 2019.

[8] H. Hosseini and R. Poovendran, "Semantic adversarial examples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1614–1619.

[9] A. Bhattad, M. J. Chong, K. Liang, B. Li, and D. A. Forsyth, "Big but imperceptible adversarial perturbations via semantic manipulation," *arXiv preprint arXiv:1904.06347*, vol. 1, no. 3, 2019.

[10] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvari, "Learning with a strong adversary," *arXiv:1511.03034*, Nov. 2015.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[12] C. Laidlaw, S. Singla, and S. Feizi, "Perceptual adversarial robustness: Defense against unseen threat models," in *ICLR*, 2021.

[13] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International conference on machine learning*. PMLR, 2020, pp. 2206–2216.

[14] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," *Advances in neural information processing systems*, vol. 31, 2018.

[15] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "Boosting robustness certification of neural networks," in *International conference on learning representations*, 2018.

[16] ——, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[17] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," *arXiv:1902.08722*, Feb. 2019.

[18] S. Dathathri, K. Dvijotham, A. Kurakin, A. Raghunathan, J. Uesato, R. R. Bunel, S. Shankar, J. Steinhardt, I. Goodfellow, P. S. Liang *et al.*, "Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5318–5331, 2020.

[19] M. N. Müller, G. Makarchuk, G. Singh, M. Püschel, and M. Vechev, "PRIMA: general and precise neural network certification via scalable convex hull approximations," *Proceedings of the ACM on Programming Languages*, vol. 6, no. POPL, pp. 1–33, jan 2022.

[20] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *arXiv:1711.00851*, Nov. 2017.

[21] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv:1801.09344*, Jan. 2018.

[22] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[23] K. Dvijotham, S. Gowal, R. Stanforth, R. Arandjelovic, B. O'Donoghue, J. Uesato, and P. Kohli, "Training verified learners with learned verifiers," *arXiv:1805.10265*, May 2018.

[24] S. Lee, W. Lee, J. Park, and J. Lee, "Towards better understanding of training certifiably robust models against adversarial examples," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[25] Y. Zhang, G. Zhang, P. Khanduri, M. Hong, S. Chang, and S. Liu, "Revisiting and advancing fast adversarial training through the lens of bi-level optimization," *arXiv:2112.12376*, Dec. 2021.

[26] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2196–2205.

[27] M. Pintor, F. Roli, W. Brendel, and B. Biggio, "Fast minimum-norm adversarial attacks through adaptive norm constraints," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[28] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv:1711.07356*, Nov. 2017.

[29] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," *arXiv:1702.01135*, Feb. 2017.

[30] R. Bunel, P. Mudigonda, I. Turkaslan, P. Torr, J. Lu, and P. Kohli, "Branch and bound for piecewise linear neural network verification," *Journal of Machine Learning Research*, vol. 21, no. 2020, 2020.

[31] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5276–5285.

[32] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018.

[33] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.

[34] Z. Lyu, C.-Y. Ko, Z. Kong, N. Wong, D. Lin, and L. Daniel, "Fastened crown: Tightened neural network robustness certificates," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5037–5044.

[35] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *arXiv:1511.04599.*, Nov. 2015.

[36] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," *arXiv:1705.08475*, May 2017.

[37] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv:1608.04644*, Aug. 2016.

[38] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4322–4330.

[39] F. E. Curtis, T. Mitchell, and M. L. Overton, "A bfgs-sqp method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles," *Optimization Methods and Software*, vol. 32, no. 1, pp. 148–181, 2017.

[40] B. Liang, T. Mitchell, and J. Sun, "NCVX: A user-friendly and scalable package for nonconvex optimization in machine learning," 2021.

[41] M. Mosbach, M. Andriushchenko, T. Trost, M. Hein, and D. Klakow, "Logit pairing methods can fool gradient-based attacks," *arXiv preprint arXiv:1810.12042*, 2018.

[42] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv:1902.06705*, Feb. 2019.

[43] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," in *European Conference on Computer Vision*. Springer, 2020, pp. 484–501.

[44] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[45] D. Bertsekas, *Nonlinear Programming 3rd Edition*. Athena Scientific, 2016.

[46] G. Pillo and M. Roma, *Large-scale nonlinear optimization*. Springer Science & Business Media, 2006, vol. 83.

[47] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[48] S. Laue, M. Mitterreiter, and J. Giesen, "Geno–generic optimization for classical machine learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[49] S. Laue, M. Blacher, and J. Giesen, "Optimization for classical machine learning problems on the gpu," *arXiv:2203.16340*, Mar. 2022.

[50] F. H. Clarke, *Optimization and nonsmooth analysis*. SIAM, 1990.

[51] A. Bagirov, N. Karmitsa, and M. M. Mäkelä, *Introduction to Nonsmooth Optimization*. Springer International Publishing, 2014.

[52] Y. Cui and J.-S. Pang, *Modern Nonconvex Nondifferentiable Optimization*. Society for Industrial and Applied Mathematics, jan 2021.

[53] J. V. Burke, F. E. Curtis, A. S. Lewis, M. L. Overton, and L. E. Simões, "Gradient sampling methods for nonsmooth optimization," *Numerical Nonsmooth Optimization*, pp. 201–225, 2020.

[54] J. M. Danskin, *The Theory of Max-Min and its Application to Weapons Allocation Problems*. Springer Berlin Heidelberg, 1967.

[55] P. Bernhard and A. Rapaport, "On a theorem of danskin with an application to a theorem of von neumann-sion," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 24, no. 8, pp. 1163–1181, apr 1995.

[56] M. Razaviyayn, T. Huang, S. Lu, M. Nouiehed, M. Sanjabi, and M. Hong, "Non-convex min-max optimization: Applications, challenges, and recent theoretical advances," *IEEE Signal Processing Magazine (Volume: 37, Issue: 5, Sept. 2020)*, Jun. 2020.

[57] P. Maini, E. Wong, and Z. Kolter, "Adversarial robustness against the union of multiple perturbation models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6640–6650.

[58] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," *arXiv preprint arXiv:2010.03593*, 2020.

[59] J. Martins and N. M. Poon, "On structural optimization using constraint aggregation," in *VI World Congress on Structural and Multidisciplinary Optimization WCSMO6, Rio de Janeiro, Brasil*. Citeseer, 2005.

[60] K.-S. Zhang, Z.-H. Han, Z.-J. Gao, and Y. Wang, "Constraint aggregation for large number of constraints in wing surrogate-based optimization," *Structural and Multidisciplinary Optimization*, vol. 59, no. 2, pp. 421–438, sep 2018.

[61] F. Domes and A. Neumaier, "Constraint aggregation for rigorous global optimization," *Mathematical Programming*, vol. 155, no. 1-2, pp. 375–401, dec 2014.

[62] Y. M. Ermoliev, A. V. Kryazhimskii, and A. Ruszczyński, "Constraint aggregation principle in convex optimization," *Mathematical Programming*, vol. 76, no. 3, pp. 353–372, mar 1997.

[63] A. C. Trapp and O. A. Prokopyev, "A note on constraint aggregation and value functions for two-stage stochastic integer programs," *Discrete Optimization*, vol. 15, pp. 37–45, feb 2015.

[64] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," *ArXiv*, vol. abs/2003.01690, 2020.

[65] ——, "Sparse and imperceivable adversarial attacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4724–4732.

[66] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[67] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity*. Chapman and Hall/CRC, may 2015.

[68] J. Wright and Y. Ma, *High-Dimensional Data Analysis with Low-Dimensional Models Principles, Computation, and Applications*. University of Cambridge ESOL Examinations, 2021.

[69] C. Studer, W. Yin, and R. G. Baraniuk, "Signal representations with minimum ℓ_∞," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, oct 2012.

[70] F. Croce and M. Hein, "Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2201–2211.

[71] Y.-L. Yu, "On decomposing the proximal map," *Advances in neural information processing systems*, vol. 26, 2013.

[72] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[73] A. S. Lewis and M. L. Overton, "Nonsmooth optimization via quasi-newton methods," *Mathematical Programming*, vol. 141, no. 1, pp. 135–163, 2013.

[74] F. H. Clarke, "Generalized gradients and applications," *Transactions of the American Mathematical Society*, vol. 205, pp. 247–262, 1975.

Appendix A
Projection onto the intersection of norm ball and box constraints

APGD for solving Eq. (1) with ℓ_p distances entails solving Euclidean projection subproblems of the form:

$$\min_{\boldsymbol{x}' \in \mathbb{R}^n} \|\boldsymbol{z} - \boldsymbol{x}'\|_2^2 \quad \text{s. t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_p \le \varepsilon,\ \boldsymbol{x}' \in [0,1]^n. \tag{14}$$

After a simple reparametrization, we have

$$\min_{\boldsymbol{\delta} \in \mathbb{R}^n} \|\boldsymbol{w} - \boldsymbol{\delta}\|_2^2 \quad \text{s. t. } \|\boldsymbol{\delta}\|_p \le \varepsilon,\ \boldsymbol{x} + \boldsymbol{\delta} \in [0,1]^n. \tag{15}$$

We focus on $p = 1, 2, \infty$ that are popular in the AR literature. In early works, a "lazy" projection scheme that sequentially projecting onto the ℓ_p ball are used. [70] has recently identified the detrimental effect of the lazy projection on the performance for $p = 1$, and derived a closed-form solution for it. Here, we prove the correctness of the sequential projection for $p = \infty$ (Lemma A.1), and discuss problems about the $p = 2$ case (Lemma A.3).

For $p = \infty$, obviously we only need to consider the individual coordinates.

Lemma A.1. *Assume $x \in [0,1]$. The unique solution for*

$$\min_{\delta \in \mathbb{R}} (w - \delta)^2 \quad \text{s. t. } |\delta| \le \varepsilon,\ x + \delta \in [0,1] \tag{16}$$

is

$$\mathcal{P}_{\infty,\text{box}} = \begin{cases} w & w \in [\max(-x,-\varepsilon), \min(1-x,\varepsilon)] \\ \max(-x,-\varepsilon) & w \le \max(-x,-\varepsilon) \\ \min(1-x,\varepsilon) & w \ge \min(1-x,\varepsilon) \end{cases}, \tag{17}$$

which agrees with the sequential projectors $\mathcal{P}_\infty \mathcal{P}_{\text{box}}$ and $\mathcal{P}_{\text{box}} \mathcal{P}_\infty$.

One can derive the one-step projection formula Eq. (17) easily once recognizing the two box constraints can be combined into one:

$$\max(-\varepsilon, -x) \le \delta \le \min(\varepsilon, 1-x). \tag{18}$$

To show the equivalence to $\mathcal{P}_\infty \mathcal{P}_{\text{box}}$ and $\mathcal{P}_{\text{box}} \mathcal{P}_\infty$, we could write down all the projectors analytically and directly verify the claimed equivalence. But that tends to be cumbersome. Here, we invoke an elegant result due to [71]. For this, we need to quickly set up the notations. For any function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, its proximal mapping $\text{Prox}_f(\boldsymbol{y})$ is defined as

$$\text{Prox}_f(\boldsymbol{y}) \doteq \arg\min_{\boldsymbol{z} \in \mathbb{R}^n} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{z}\|_2^2 + f(\boldsymbol{z}). \tag{19}$$

When f is the indicator function $\imath_C$ for a set C defined as

$$\imath_C(\boldsymbol{z}) = \begin{cases} 0 & \boldsymbol{z} \in C \\ \infty & \text{otherwise} \end{cases}, \tag{20}$$

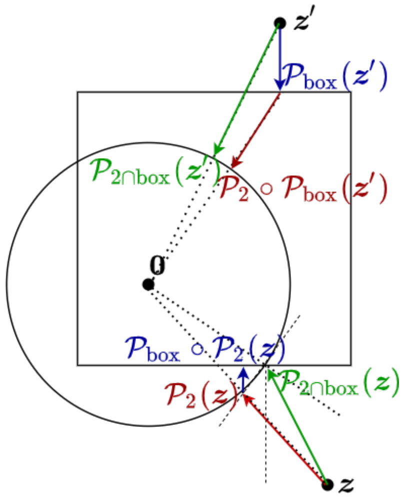
$\text{Prox}_f(\boldsymbol{y})$ is the Euclidean projector $\mathcal{P}_C(\boldsymbol{y})$. For two closed proper convex functions f and g, [71] studies when $\text{Prox}_{f+g} = \text{Prox}_f \circ \text{Prox}_g$. If f and g are two set indicator functions, this exactly asks when the sequential projector is equivalent to the true projector.

Theorem A.2 (adapted from Theorem 2 of [71])**.** *If $f = \imath_C$ for a closed convex set $C \subset \mathbb{R}$, $\text{Prox}_f \circ \text{Prox}_g = \text{Prox}_{f+g}$ for all closed proper convex functions $g : \mathbb{R} \to \mathbb{R} \cup \{\pm\infty\}$.*

The equivalence of projectors we claim in Lemma A.1 follows by setting $f = \imath_\infty$ and $g = \imath_{\text{box}}$, and vise versa.

For $p = 2$, the sequential projectors are not equivalent to the true projector in general, although empirically we observe that $\mathcal{P}_2\mathcal{P}_{\text{box}}$ is a much better approximation than $\mathcal{P}_{\text{box}}\mathcal{P}_2$. The former is used in the APGD algorithm of current `AutoAttack`.

Lemma A.3. *Assuming $\boldsymbol{x} \in [0,1]^n$. When $p = 2$, the projector for Eq.* (15) *$\mathcal{P}_{2,\text{box}}$ does not agree with the sequential projectors $\mathcal{P}_2\mathcal{P}_{\text{box}}$ and $\mathcal{P}_{\text{box}}\mathcal{P}_2$ in general. But both $\mathcal{P}_2\mathcal{P}_{\text{box}}$ and $\mathcal{P}_{\text{box}}\mathcal{P}_2$ always find feasible points for the projection problem.*



$\boldsymbol{z}'$
$\mathcal{P}_{\text{box}}(\boldsymbol{z}')$
$\mathcal{P}_{2\cap\text{box}}(\boldsymbol{z}')$
$\mathcal{P}_2 \circ \mathcal{P}_{\text{box}}(\boldsymbol{z}')$
$\boldsymbol{0}$
$\mathcal{P}_{\text{box}} \circ \mathcal{P}_2(\boldsymbol{z})$
$\mathcal{P}_{2\cap\text{box}}(\boldsymbol{z})$
$\mathcal{P}_2(\boldsymbol{z})$
$\boldsymbol{z}$

Fig. 7. Illustration of the problem with the sequential projectors when $p = 2$. In general, neither of the sequential projectors produces the right projection.

Proof. For the nonequivalence, we present a couple of counter-examples in Fig. 7. Note that the point $\boldsymbol{z}$ is inside the normal cone of the bottom right corner point of the intersection: $\{\boldsymbol{\delta} \in \mathbb{R}^2 : \|\boldsymbol{\delta}\|_2 \le \varepsilon\} \cap \{\boldsymbol{\delta} \in \mathbb{R}^2 : \boldsymbol{x} + \boldsymbol{\delta} \in [0,1]^2\}$.

For the feasibility claim, note that for any $\boldsymbol{y} \in \mathbb{R}^n$

$$\mathcal{P}_2(\boldsymbol{y}) = \begin{cases} \varepsilon \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2} & \|\boldsymbol{y}\|_2 \ge \varepsilon \\ \boldsymbol{y} & \text{otherwise} \end{cases}, \tag{21}$$

and for any $y \in \mathbb{R}$,

$$\mathcal{P}_{\text{box}}(y) = \begin{cases} 1-x & y \ge 1-x \\ y & -x < y < 1-x \\ -x & y \le -x \end{cases}, \tag{22}$$

and $\mathcal{P}_{\text{box}}(\boldsymbol{y})$ acts on any $\boldsymbol{y} \in \mathbb{R}^n$ elementwise. For any $\boldsymbol{y}$ inside the ℓ_2 ball,

$$\begin{aligned} \|\mathcal{P}_{\text{box}}(\boldsymbol{y})\|_2 &= \|\mathcal{P}_{\text{box}}(\boldsymbol{y}) - \mathcal{P}_{\text{box}}(\boldsymbol{0})\|_2 \\ &\le \|\boldsymbol{y} - \boldsymbol{0}\|_2 = \|\boldsymbol{y}\|_2 \le \varepsilon \end{aligned} \tag{23}$$

due to the contraction property of projecting onto convex sets. So $\mathcal{P}_{\text{box}}\mathcal{P}_2(\boldsymbol{y})$ is feasible for any $\boldsymbol{y} \in \mathbb{R}^n$. Now for any $\boldsymbol{y}$ inside the box:

- if $\|\boldsymbol{y}\|_2 < \varepsilon$, $\mathcal{P}_2(\boldsymbol{y}) = \boldsymbol{y}$ and so $\mathcal{P}_2(\boldsymbol{y})$ remains in the box;
- if $\|\boldsymbol{y}\|_2 \ge \varepsilon, \mathcal{P}_2(\boldsymbol{y}) = \varepsilon \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2}$. Since $\varepsilon/\|\boldsymbol{y}\|_2 \in [0,1]$, $\boldsymbol{P}_2(\boldsymbol{y})$ shrinks each component of $\boldsymbol{y}$ but keeps their original signs. Thus $\boldsymbol{P}_2(\boldsymbol{y})$ remains in the box if $\boldsymbol{y}$ is in the box.

We conclude that $\mathcal{P}_2\mathcal{P}_{\text{box}}(\boldsymbol{y})$ is feasible for any $\boldsymbol{y}$, completing the proof. ☐

Appendix B
Sketch of the BFGS-SQP algorithm in GRANSO

GRANSO is among the first optimization solvers targeting general nonsmooth NCVX problems with nonsmooth constraints [39].

The key algorithm of GRANSO package is a sequential quadratic method that employs a quasi-Newton method, Broyden-Fletcher-Goldfarb-Shanno (BFGS) [44], and an exactly penalty function.

Penalty-SQP method: alternative search direction can be calculated from the following QP

$$\begin{aligned} \min_{\boldsymbol{d}\in\mathbb{R}^n, \boldsymbol{s}\in\mathbb{R}^p} \ & \mu(f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\intercal \boldsymbol{d}) + \boldsymbol{e}^\intercal \boldsymbol{s} + \frac{1}{2}\boldsymbol{d}^\intercal \boldsymbol{H}_k \boldsymbol{d} \\ \text{s.t.}\ & c(\boldsymbol{x}_k) + \nabla c(\boldsymbol{x}_k)^\intercal \boldsymbol{d} \le \boldsymbol{s}, \quad \boldsymbol{s} \ge 0 \end{aligned} \tag{24}$$

The dual of problem (Eq. (24)) is used in GRANSO package:

$$\begin{aligned} \max_{\boldsymbol{\lambda}\in\mathbb{R}^p} \quad & \mu f(\boldsymbol{x}_k) + c(\boldsymbol{x}_k)^\intercal \boldsymbol{\lambda} \\ & - \frac{1}{2}(\mu\nabla f(\boldsymbol{x}_k) + \nabla c(\boldsymbol{x}_k)\boldsymbol{\lambda})^\intercal \boldsymbol{H}_k^{-1}(\mu\nabla f(\boldsymbol{x}_k) + \nabla c(\boldsymbol{x}_k)\boldsymbol{\lambda}) \\ \text{s.t.}\ & 0 \le \boldsymbol{\lambda} \le \boldsymbol{e} \end{aligned} \tag{25}$$

This QP has only simple box constraints, which can be easily handled by many popular QP solvers like OSQP (ADMM based algorithm) [72]. By solving the dual problem (Eq. (25)), the primal solution $\boldsymbol{d}$ can be recovered from the dual solution $\boldsymbol{\lambda}$:

$$\boldsymbol{d} = -\boldsymbol{H}_k^{-1}(\mu\nabla f(\boldsymbol{x}_k) + \nabla c(\boldsymbol{x}_k)\boldsymbol{\lambda}) \tag{26}$$

Searching direction calculated at each step controls the tradeoff between minimizing the objective and moving towards feasible region. To measure the how much violence the current searching direction will give, a linear model of constraint violation is used:

$$l(\boldsymbol{d};\boldsymbol{x}_k) := \|\max\{c(\boldsymbol{x}_k) + \nabla c(\boldsymbol{x}_k)^\intercal \boldsymbol{d}, \boldsymbol{0}\}\|_1 \tag{27}$$

To dynamically set the penalty parameter, the steering strategy algorithm is used:

Algorithm 1 $[\boldsymbol{d}_k, \boldsymbol{\mu}_{\text{new}}] =$ `sqp_steering`$(\boldsymbol{x}_k, \boldsymbol{H}_k, \boldsymbol{\mu})$

```
Require: $\boldsymbol{x}_k, \boldsymbol{H}_k, \mu$ at current iteration
Require: constants $c_v \in (0,1), c_\mu(0,1)$
1: Calculate $\boldsymbol{d}_k$ from (Eq. (26)) and (Eq. (25)) with $\mu_{\text{new}} = \mu$
2: if $l_\delta(\boldsymbol{d}_k;\boldsymbol{x}_k) < c_v v(\boldsymbol{x}_k)$ then
3:     Calculate $\tilde{\boldsymbol{d}}_k$ from (Eq. (26)) and (Eq. (25)) with $\mu = 0$
4:     while $l_\delta(\boldsymbol{d}_k;\boldsymbol{x}_k) < c_v l_\delta(\tilde{\boldsymbol{d}}_k;\boldsymbol{x}_k)$ do
5:         $\mu_{new} := c_\mu \mu_{\text{new}}$
6:         Calculate $\boldsymbol{d}_k$ from (Eq. (26)) and (Eq. (25)) with $\mu = \mu_{\text{new}}$
7:     end while
8: end if
9: return $\boldsymbol{d}_k, \boldsymbol{\mu}_{\text{new}}$
```

For nonsmooth problem, it's usually hard to find a reliable stopping criterion as the norm of the gradient won't decrease when approaching the minimizer. GRANSO uses an alternative stopping strategy, which is based on the idea of gradient sampling [73] [53].

Define the neighboring gradient information (from the l most recent iterates) as:

$$\begin{aligned} \boldsymbol{G} &:= \left[\nabla f(\boldsymbol{x}_{x_{k+1-l}}) \dots \nabla f(\boldsymbol{x}_k)\right] \\ \boldsymbol{J}_i &:= \left[\nabla c_i(\boldsymbol{x}_{x_{k+1-l}}) \dots \nabla c_i(\boldsymbol{x}_k)\right], \quad i \in \{1, \dots, p\} \end{aligned} \tag{28}$$

Augment the QP (Eq. (24)) and its dual (Eq. (25)) in the steering strategy, we can obtain the augmented dual problem:

$$\begin{aligned} &\max_{\boldsymbol{\sigma}\in\mathbb{R}^l, \boldsymbol{\lambda}\in\mathbb{R}^{pl}} \sum_{i=1}^{p} c_i(\boldsymbol{x}_k)\boldsymbol{e}^\intercal \boldsymbol{\lambda}_i \\ &- \frac{1}{2}\begin{bmatrix}\boldsymbol{\sigma}\\ \boldsymbol{\lambda}\end{bmatrix}^\intercal \left[\boldsymbol{G}, \boldsymbol{J}_1, \dots, \boldsymbol{J}_p\right]^\intercal \boldsymbol{H}_k^{-1} \left[\boldsymbol{G}, \boldsymbol{J}_1, \dots, \boldsymbol{J}_p\right] \begin{bmatrix}\boldsymbol{\sigma}\\ \boldsymbol{\lambda}\end{bmatrix} \\ &\text{s. t. } \boldsymbol{0} \le \boldsymbol{\lambda}_i \le \boldsymbol{e}, \quad \boldsymbol{e}^\intercal\boldsymbol{\sigma} = \mu, \quad \boldsymbol{\sigma} \ge \boldsymbol{0} \end{aligned} \tag{29}$$

By solving QP (Eq. (29)), we can obtain $\boldsymbol{d}_\diamond$:

$$\boldsymbol{d}_\diamond = \boldsymbol{H}_k^{-1} \left[\boldsymbol{G}, \boldsymbol{J}_1, \dots, \boldsymbol{J}_p\right] \begin{bmatrix}\boldsymbol{\sigma}\\ \boldsymbol{\lambda}\end{bmatrix} \tag{30}$$

If the norm of $\boldsymbol{d}_\diamond$ is sufficiently small, the current iteration can be viewed as near a small neighborhood of a stationary point.

Algorithm 2 $[\boldsymbol{x}_*, f_*, \boldsymbol{v}_*] =$ `bfgs_sqp`$(f(\cdot), \boldsymbol{c}(\cdot), \boldsymbol{x}_0, \mu_0))$

```
Require: $f, \boldsymbol{c}, \boldsymbol{x}_0, \mu_0$
Require: constants $\tau_\diamond, \tau_v$
 1: $\boldsymbol{H}_0 := \boldsymbol{I},\ \mu := \mu_0$
 2: $\phi(\cdot) = \mu f(\cdot) + v(\cdot)$
 3: $\nabla\phi(\cdot) = \mu\nabla f(\cdot) + \sum_{i\in\mathcal{P}} \nabla c_i(\cdot)$
 4: $v(\cdot) = \|\max\{c(\cdot), 0\}\|1$
 5: $\phi_0 := \phi(\boldsymbol{x}_0;\mu), \nabla\phi_0 := \nabla\phi(\boldsymbol{x}_0;\mu), v_0 := v(\boldsymbol{x}_0)$
 6: for $k = 0, 1, 2, \dots$ do
 7:     $[\boldsymbol{d}_k, \hat{\mu}] :=$ sqp_steering$(\boldsymbol{x}_k, \boldsymbol{H}_k, \boldsymbol{\mu})$
 8:     if $\hat{\mu} < \mu$ then
 9:         $\mu := \hat{\mu}$
10:         $\phi_k := \phi(\boldsymbol{x}_k;\mu), \nabla\phi_k := \nabla\phi(\boldsymbol{x}_k;\mu), v_k := v(\boldsymbol{x}_k)$
11:     end if
12:     $[\boldsymbol{x}_{k+1}, \phi_{k+1}, \nabla\phi_{k+1}, v_{k+1}]$ :=
        Armijo_Wolfe$(\boldsymbol{x}_k, \phi_k, \nabla\phi_k, \phi(\cdot), \nabla\phi(\cdot))$
13:     Get $\boldsymbol{d}_\diamond$ from (Eq. (30)) and (Eq. (29))
14:     if $\|\boldsymbol{d}_\diamond\|2 < \tau_\diamond$ and $v_{k+1} < \tau_v$ then
15:         break
16:     end if
17:     BFGS update $\boldsymbol{H}_{k+1}$
18: end for
19: return $\boldsymbol{x}_*, f_*, \boldsymbol{v}_*$
```

Appendix C
Danskin's theorem and min-max optimization

In this section, we discuss the importance of computing a good solution for the inner maximization problem when applying first-order methods for AT, i.e., solving Eq. (2).

Consider the minimax problem

$$\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) \doteq \left[\max_{\boldsymbol{x}'\in\Delta} \ h(\boldsymbol{\theta}, \boldsymbol{x}')\right], \tag{31}$$

where we assume the function h is locally Lipschitz continuous. In order to apply first-order methods to solve Eq. (31), one needs to evaluate a (sub)gradient of g at any given $\boldsymbol{\theta}$. If $h(\boldsymbol{\theta}, \boldsymbol{x}')$ is smooth in $\boldsymbol{\theta}$, one can invoke Danskin's theorem for such an evaluation (see for example [11, Appendix A]). However, in DL applications with nonsmooth activations or losses, $h(\boldsymbol{\theta}, \delta)$ is not differentiable in $\boldsymbol{\theta}$, and hence a general version of Danskin's theorem is needed.

To proceed, we first introduce a few basic concepts in nonsmooth analysis; general background can be found in [50], [51], [52]. For a locally Lipschitz continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$, define its Clarke directional derivative at $\bar{\boldsymbol{z}} \in \mathbb{R}^n$ in any direction $\boldsymbol{d} \in \mathbb{R}^n$ as

$$\varphi^\circ(\bar{\boldsymbol{z}}; \boldsymbol{d}) \doteq \limsup_{t\downarrow 0, \boldsymbol{z}\to\bar{\boldsymbol{z}}} \frac{\varphi(\boldsymbol{z} + t\boldsymbol{d}) - \varphi(\boldsymbol{z})}{t}.$$

We say φ is Clarke regular at $\bar{\boldsymbol{z}}$ if $\varphi^\circ(\bar{\boldsymbol{z}}; \boldsymbol{d}) = \varphi'(\bar{\boldsymbol{z}}; \boldsymbol{d})$ for any $\boldsymbol{d} \in \mathbb{R}^n$, where $\varphi'(\bar{\boldsymbol{z}}; \boldsymbol{d}) \doteq \lim_{t\downarrow 0} \frac{1}{t}(\varphi(\bar{\boldsymbol{z}} + t\boldsymbol{d}) - \varphi(\bar{\boldsymbol{z}}))$ is the usual one-sided directional derivative. The Clarke subdifferential of φ at $\bar{\boldsymbol{z}}$ is defined as

$$\partial\varphi(\bar{\boldsymbol{z}}) \doteq \{\boldsymbol{v} \in \mathbb{R}^n : \varphi^\circ(\bar{\boldsymbol{z}}; \boldsymbol{d}) \ge \boldsymbol{v}^\intercal\boldsymbol{d}\}.$$

The following result has its source in [74, Theorem 2.1]; see also [52, Section 5.5].

Theorem C.1. *Assume that Δ in Eq. (31) is a compact set, and the function h satisfies*

1) *h is jointly upper semicontinuous in $(\boldsymbol{\theta}, \boldsymbol{x}')$;*
2) *h is locally Lipschitz continuous in $\boldsymbol{\theta}$, and the Lipschitz constant is uniform in $\boldsymbol{x}' \in \Delta$;*
3) *h is directionally differentiable in $\boldsymbol{\theta}$ for all $\boldsymbol{x}' \in \Delta$;*

If h is Clarke regular in $\boldsymbol{\theta}$ for all $\boldsymbol{\theta}$, and $\partial_{\boldsymbol{\theta}} h$ is upper semicontinuous in $(\boldsymbol{\theta}, \boldsymbol{x}')$, we have that for any $\bar{\boldsymbol{\theta}}$

$$\partial g(\bar{\boldsymbol{\theta}}) = \operatorname{conv}\{\partial h(\bar{\boldsymbol{\theta}}, \boldsymbol{x}') : \boldsymbol{x}' \in \Delta^*(\bar{\boldsymbol{\theta}})\}, \tag{32}$$

where $\operatorname{conv}(\cdot)$ denotes the convex hull of a set, and $\Delta^*(\bar{\boldsymbol{\theta}})$ is the set of all optimal solutions of the inner maximization problem at $\bar{\boldsymbol{\theta}}$.

The above theorem indicates that in order to get an element from the subdifferential set $\partial g(\bar{\boldsymbol{\theta}})$, we need to get at least one **optimal** solution $\boldsymbol{x}' \in \Delta^*(\bar{\boldsymbol{\theta}})$. A suboptimal solution to the inner maximization problem may result in a useless direction for the algorithm to proceed. To illustrate this, let us consider a simple one-dimensional example

$$\min_{\theta} g(\theta) := \left[\max_{-1\le x'\le 1} \ \max(\theta x', 0)^2\right],$$

which corresponds to a one-layer neural network with one data point $(0, 0)$, the ReLU activation function and squared loss. Starting at $\theta_0 = 1$, we get the first inner maximization problem $\max_{-1\le x'\le 1} \max(x', 0)^2$. Although its global optimal solution is $x'_* = 1$, the point $x' = 0$ is a stationary point satisfying the first-order optimality condition. If the latter point is mistakenly adopted to compute an element in $\partial g(\theta^0)$, it would result in a zero direction so that the overall gradient descent algorithm cannot proceed.